The Effect of Inexact Line Search On Conjugate Gradient Methods

by

Hattan Zain Al-Abdin Tawfiq

A Thesis Presented to the

FACULTY OF THE COLLEGE OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

MATHEMATICS

September, 1991

Order Number 1354100

The effect of inexact line search on conjugate gradient methods

Tawfiq, Hattan Zain Al-Abdin, M.S.

King Fahd University of Petroleum and Minerals (Saudi Arabia), 1991

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

THE EFFECT OF INEXACT LINE SEARCH ON CONJUGATE GRADIENT METHODS

BY

HATTAN ZAIN AL-ABDIN TAWFIQ

A Thesis Presented to the

FACULTY OF THE COLLEGE OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

In

MATHEMATICS

SEPTEMBER 1991

KING FAHD UNIVERSTY

OF PETROLEUM AND MINERALS

DHAHRAN 31261, SAUDI ARABIA

COLLEGE OF GRADUATE STUDIES

This thesis, written by **HATTAN ZAIN AL-ABDIN TAWFIQ** under the direction of his Thesis Advisor and approved by his Thesis Committee, has been presented to and accepted by the Dean of the College of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE** *IN MATHEMATICS.*

Thesis Committee

Thesis Advisor

Member

Member DEC-14, 91

Department Chairman

Dean, College of Graduate Studies

Date 15-12-91

ACKNOWLEDGEMENT

All praise be to ALLAH, the lord of the world, the Almighty, with whose gracious help it was possible to accomplish this work. May peace and blessings be upon Mohammad the last of the Messengers.

Then I would like to express my appreciation to my supervisor Dr. M. Bokhari for introducing me to the subject and for his continuous help and encouragement at all stages of preparation of this Thesis. I also would like to thank the other members of my Thesis commitee Drs. F. Badawai and M. Ben-Daya.

Finally, I thank King Fahd University of Petroleum and Minerals for supporting this research.

TABLE OF CONTENTS

ABSTRACT

Line search is a major component in the structure of the conjugate gradient method to solve unconstraind optimization problems. But its exact implementation to nonquadratic functions is unlikely in most of the cases. This thesis deals with a self contained study of inexact line search and its effect on the convergence of certain modifications and extensions of the conjugate gradient method. We describe in detail various algorithms due to these extensions and apply them to some of the standard test functions.

خلاصة الرساله

البحث الخطي مركب مهم في بنية طريقة الميل المترافق لحل مسائل الإبتمايزييشن غير المقيد ه . لكن التطبيق الدقيق لهذه الطريقه في الدوال من غير الدرجه الثانيه غير ممكن في معظم الحالات.

هذه الرساله تحتوي على دراسه شامله للبحث الخطي غير الدقيق وتاثيره على تقارب بعض الطرق المطوره والمحسنه لطريقة الميل المترافق. سوف نشرح بالتفصيل عدة خوارزميات لهذه الطرق المطوره ثم نطبقها على بعض دوال الاختبار النموذجيه.

CHAPTER 1

INTRODUCTION

One of the fundamental problems in optimization is that of developing computational procedures for finding extremum of a real-valued function. If the variables of the function are not subject to any restrictions then the problem is referred to as unconstrained optimization problem in the literature.

Many algorithms have been designed to iteratively solve unconstrained optimization problems of the form:

$$\text{minimize } f(x) \qquad , x \in \Re^n,$$

where f is a real-valued function.

Although these algorithms vary substantially in their motivation, application, and analysis, they are all iterative descent algorithms.

The underlying structure for most of such algorithms is as follows:

Step 1. Start at some initial point.

Step 2. Find a direction of movement according to a predetermined rule.

Step 3. Find a new point which is relative minimum of the obgective function on the line in the direction found in step 2.

function on the line in the direction found in step 2.

Step 4. Replace the point in step 1 by the new point.

Step 5. Repeat steps 2 to 4 if needed.

Next, we define some of the terminologies which will be used frequently [9]:

Definition 1.1 An algorithm that generates a series of points where each point is calculated by using the points preceding it, is called *iterative algorithm.*

Definition 1.2 At a new point generated by an algorithm if the corresponding value of an objective function decreases with respect to the preceding iterates then this algorithm is called *descent algorithm.*

Definition 1.3 The process of minimizing a given function along straight lines is known as *line search.*

Definition 1.4 A sequance $\{ x_n \}$ converges to a point x^* if

$$\lim_{k \to \infty} \| x_k - x^* \| = 0$$

Definition 1.5 If for arbitrary starting point the algorithm is guaranteed to generate a sequance of points converging to a solution, then the algorithm is said to be ***globally convergent.***

Remark It is known that the direction of the steepest ascent of a function f at a point x is given by the gradient vector $\nabla f(x)$ of f at x. For the sake of simplicity we shall write $\nabla f = g$. Therefore, the vector $(-g)$ points in the direction of steepest descent of f.

Based on the notion of line search, one of the oldest and most obvious methods for obtaining a minimum point x^* of f is the *method of steepest descent.* In this method we start with an initial point x_1 and minimize f successively along lines in the directions of steepest descent. Although the method of steepest descent is useful for well-conditioned problems, experience has shown that the method is extremely slow in most of the cases.

Because of its simplicity and stability the method of steepest descent is often a fundamental component of more elaborate techniques. An important method which contains steepest descent method as one of its components is

the method of *conjugate gradient*.

The objective of this thesis is twofold. The first one is to discuss issues related to the subproblem of line search. In particular, we present the different conditions on inexact line search leading to convergent algorithms. The second one is to study the impact of inexact line search on various conjugate gradient methods.

The next chapter deals with the study of the subproblem of line search. There, we also explain the notion of inexact line search and discuss certain conditions which make it effective when coupled with conjugate gradient techniques.

In Chapter 3 we shall describe in detail the conjugate gradient method and its modifications due to Fletcher-Reeves, Polak-Ribiere, Shanno, and Touati Ahmed-Storey. We shall explain these modified algorithms in the light of inexact line search and also their global convergence.

Finally, in Chapter 4 we test the algorithms using standard functions like Rosenbrock test function and Powell test function and another nonquadratic nonstandard function. For each of these function, we set up a separate table in which all the methods are compared after detailed computer programming.

CHAPTER 2

LINE SEARCH

2.1 INTRODUCTION

As we have seen in the first chapter, line search is an important step in the method of steepest descent as well as in the method of conjugate gradient. Numerous line search algorithms have been proposed over the years , and a good choice is important since it can have a considerable effect on the performance of the method in which it is embedded.

A line search algorithm is an iterative method which generates a sequence of estimates. The sequence terminates when an iterate satisfying some standard conditions for an acceptable point is located.

In the next section of this chapter we shall give some of the most known line search algorithms. We shall study inexact line search in section 3 and give the derivation of the standard conditions on inexact line search.

In the last section we describe a modified quadratic algorithm which will be used in the further developments.

2.2 LINE SEARCH ALGORITHMS

In the following subsections, we shall describe three important line search algorithms: Fibonacci search, Newton's Method and Quadratic Fit [9]. We end the section by an example in order to compare the performance of theses three algorithms.

2.2.1 FIBONACCI SEARCH

The method of Fibonacci search searches for the minimum of a function ϕ over a closed interval $[c_1, c_2]$ by measuring its values at a certain number of points. The method is applicable if ϕ has a single relative minimum, say x^*, in $[c_1, c_2]$. In this method, the underlying idea is to squeeze $[c_1, c_2]$ in such a manner that x^* is enclosed in the resulting interval of reduced length. We set

$$d_1 = c_2 - c_1$$

which is the initial width of uncertainty interval and define

$$d_k = \text{width of uncertainty interval after } k \text{ measurements.}$$

If we decide at hand about N, the number of measurements of ϕ, then

the width of $[c_1, c_2]$ will be reduced successively to

$$d_k = (\frac{F_{N-k+1}}{F_N})d_1 \quad k = 2, 3, \ldots, N.$$

Here $\{ F_k \}$ represents the Fibonacci sequence which is generated by the recurrence relation:

$$F_k = F_{k-1} + F_{k-2}$$

$$F_0 = 1, \ F_1 = 1.$$

The Fibonacci sequence, in fact, turns out to be $1, 1, 2, 3, 5, 8, \ldots$. Thus, the width of $[c_1, c_2]$ after taking into account N measurements of ϕ reduces to $\frac{d_1}{F_N}$.

The method of Fibonacci search is applied as follows:

Step 1. Decide N, the number of measurements to be performed.

Step 2. For $k = 1, 2, \ldots, N-2$, fix two points x'_k, x''_k in the interval $[c'_k, c''_k]$ so that x'_k and x''_k are at a distance of d_{k+1} respectively from c'_k and c''_k.

Step 3. Replace c'_k by x'_k if $\phi(x'_k) > \phi(x''_k)$. Otherwise replace c''_k by x''_k.

In this way, we have a new interval $[c_{k+1}, c''_{k+1}]$.

Step 4. If $k < N-2$, go to step 1. Otherwise go to step 5.

Step 5. Fix the mid point c of $[c'_{N-1}, c''_{N-2}]$. Choose two points x' and x''

fairly close to c on its left and right side respectively.

Step 6. Replace c'_{N-1} by x' if $\phi(x') > \phi(x'')$. Otherwise replace c''_{N-1} by x''.

(The resulting interval has an approximate width of $(1/2)d_{N-1} = d_N$).

Regarding step 2 it may be noted that one of the two points x' and x'' is carried forward from the information obtained at certain i^{th} stage where $i < k$ and $k \geq 2$. Therefore, we measure ϕ at $N-2$ points of the form x'_k or x''_k, $k = 1, 2, \ldots, N-2$. Also two additional measurements are made at the final stage as described in step 5 and step 6.

Remark . The main disadvantage of this method is that the number of iterations has to be worked out in advance . This is inconvenient if the minimization requires a specified reduction in function value rather than interval.

2.2.2 NEWTON'S METHOD

Newton's method provides an iterative procedure to determine a critical point of a given function. This method is applicable only to a class of twice differentiable functions. Its description for a real valued function ϕ depending on one variable is as follows:

Step 1. Start at an initial point x_0.

Step 2. For $k = 0, 1, 2, \ldots$ find

$$x_{k+1} = x_k - \frac{\phi'(x_k)}{\phi''(x_k)}.$$

The sequence $\{ x_k \}$ generated by the above recurrsive relation is known as Newton's sequence. This sequence definitely converges to x^*, a critical point of ϕ, if ϕ is twice continuously differentiable, $\phi''(x^*) \neq 0$ and x_0 is sufficiently close to x^*.

In fact, x_{k+1} is the critical point of the quadratic function which at x_k agrees with ϕ up to second derivatives.

Remark While in the Fibonacci search, we require only the evaluation of the function, we note that the evaluation of the first and the second derivative of the function is required in the Newton's method. The main advantage of this method is that it has order two convergence [9].

2.2.3 QUADRATIC FIT

In order to implement Newton's method, we have already noticed that an information about the second derivative of the function ϕ is required at each iteration. There is another iterative procedure known as quadratic

fit algorithm for locating the minimum of ϕ which does not demand any derivative information. We describe the algorithm below for a function ϕ having single relative minimum x^*:

Step 1. Choose a three-point pattern (x_1, x_2, x_3) such that

(i) $x_1 < x_2 < x_3$.

(ii) $\phi(x_1) \geq \phi(x_2)$ and $\phi(x_2) \leq \phi(x_3)$.

Step 2. For $i, j = 1, 2, 3, \ldots$ compute

$$a_{ij} = x_i - x_j$$

$$b_{ij} = x_i^2 - x_j^2.$$

Step 3. Compute

$$x_4 = (1/2)\frac{b_{23}\phi(x_1) + b_{31}\phi(x_2) + b_{12}\phi(x_3)}{a_{23}\phi(x_1) + a_{31}\phi(x_2) + a_{12}\phi(x_3)}$$

Step 4. If $x_4 \in (x_2, x_3)$, go to step 5. Else go to step 6.

Step 5. Go to step 1 and use the following three point pattern:

$$(\overline{x}_1, \overline{x}_2, \overline{x}_3) = \begin{cases} (x_2, x_4, x_3) & \text{if } \phi(x_4) \leq \phi(x_2) \\ (x_1, x_2, x_4) & \text{otherwise} \end{cases}$$

Step 6. Go to step 1 and use the following three point pattern:

$$(\overline{x}_1, \overline{x}_2, \overline{x}_3) = \begin{cases} (x_1, x_4, x_2) & \text{if } \phi(x_4) \leq \phi(x_1) \\ (x_4, x_2, x_3) & \text{otherwise} \end{cases}$$

It is intersting to note that the point x_4 determined at step 3 is the minima of the quadratic function which agrees to ϕ at the points x_1, x_2, x_3. Moreover, the convergence of this iterative sheme is guarentteed to x^* if ϕ is twice continuously differentiable [4].

To see the performance of the three algorithms we consider the following example [14]:

Example Let $f(x) = e^{-x} + x^2, \quad x \in \Re$.

The results are shown in the following table where the initial interval is $[0, 1]$ and the initial point is 0.3.

k	Fibonacci		Newton		Quadratic	
	x	$f(x)$	x	$f(x)$	x	$f(x)$
1	0.3819	0.8284	0.300	0.83082	0.300	0.8308
2	0.2361	0.8454	0.3514	0.827184	0.3618	0.8273
3	0.3819	0.8284	0.3517	0.827184	0.3528	0.8272
4	0.3264	0.8281			0.3518	0.827184
5	0.2917	0.8321			0.3517	0.827184
6	0.2917	0.8281				
7	0.3472	0.8272				
8	0.3403	0.8274				

In the above table, the numbers corresponding to Fibonacci search are taken from [14]. The details of computational work may be found in the appendix. Note that Newton's method requires first and second derivative information. Also the starting point should not be far away from the optimal solution for this method to converge.

2.3 INEXACT LINE SEARCH

It is important to note that accurate line searches are expensive to carry out, and there is also the nuisance that the exact minimizer may not exist. As a matter of fact, it is often desirable to sacrify accuracy at the cost of global convergence in the line search routine[4]. This is done generally by terminating the search procedure before it converges. This phenomenon is known as *inexact line search* .

In order to weaken the line search tolerance some researchers used the descent property to force a decrease $f_{k+1} < f_k$ in the objective function on each iteration. However, requiring a decrease in f does not ensure global convergence. Thus there were doubts about the stability of this approach. Due to this fact and the development of various techniques to implement line search, certain conditions for terminating the line search were established in the past. These conditions ensure the global convergence at the cost of low accuracy [4].

Remark: It may be noted that a line L through a point x_1 in the direction of a nonzero vector d_1 can be expressed parametrically by the equation $x = x_1 + \alpha d_1$, where α is a parameter. If we restrict a differentiable function

f on the points of L then f may be regarded as a function of α, i.e.,

$$\phi(\alpha) = f(x_1 + \alpha d_1).$$

Therefore, a point $x^* = x_1 + \alpha^* d_1$ will be a critical point of f on L if $\phi'(\alpha^*) = 0$, i.e. $g_*^T d_1 = 0$ where g_* is the gradient of f at x^*.

Now we proceed to the description of various conditions required for the practicable implementation of inexact line search.

Let $\overline{\alpha}_k = \text{glb}\ \{\ \alpha : f(x_k + \alpha d_k) = f(x_k)\ \}$. Suppose that $\alpha_k \in [0, \overline{\alpha}_k]$ be a step size determined in an inexact line search along the line $\{\ x_k + \alpha d_k : \alpha \in \Re\ \}$. If α_k is fairly close to either $\overline{\alpha}_k$ or zero then one may expect a negligible reduction in f. Therefore, in order to develop a set of conditions for the implementation of inexact line search, we must take care of the following points:

(i) The step size α_k should not be close to the end points of the interval $[0, \overline{\alpha}_k]$.

(ii) α_k should be determined in a finite number of steps.

(iii) The conditions should not exclude the exact minimizer of

$f(x_k + \alpha d_k)$ along the line whenever f is a quadratic

function with positive curvature.[4]

Let us define the function $\phi(\alpha) = f(x_k + \alpha d_k)$ so that we have

$$\phi(0) = f(x_k) \quad , \quad \phi'(\alpha) = f'(x_k + \alpha d_k)^T . d_k$$

The linear approximation of $\phi(\alpha)$ is given by

$$\phi(\alpha) \approx \phi(0) + \alpha g_k^T d_k$$

where g_k is the gradient of f at x_k. In terms of the function f the above relation will be :

$$f(x_{k+1}) \approx f(x_k) + \alpha g_k^T d_k$$

where $x_{k+1} = x_k + \alpha d_k$. In order to have a decrease in f we must have

$$\alpha g_k^T d_k < 0.$$

From the descent property which is satisfied by all descent methods (as we shall see later) we get:

$$\alpha > 0$$

Goldstein conditions In 1965, Goldstein [6] gave two conditions on α to

achieve the above requirements (i.e. α is not close to the extremes of the interval and α_k is positive) :

I - $\phi(\alpha) \leq \phi(0) + \alpha\rho\phi'(0)$

This condition excludes the right-hand extreme .

II- $\phi(\alpha) \geq \phi(0) + \alpha(1-\rho)\phi'(0)$

This condition excludes the left-hand extreme .

In both conditions (I) and (II) given above, $\rho \in (0, \frac{1}{2})$ is a fixed parameter.

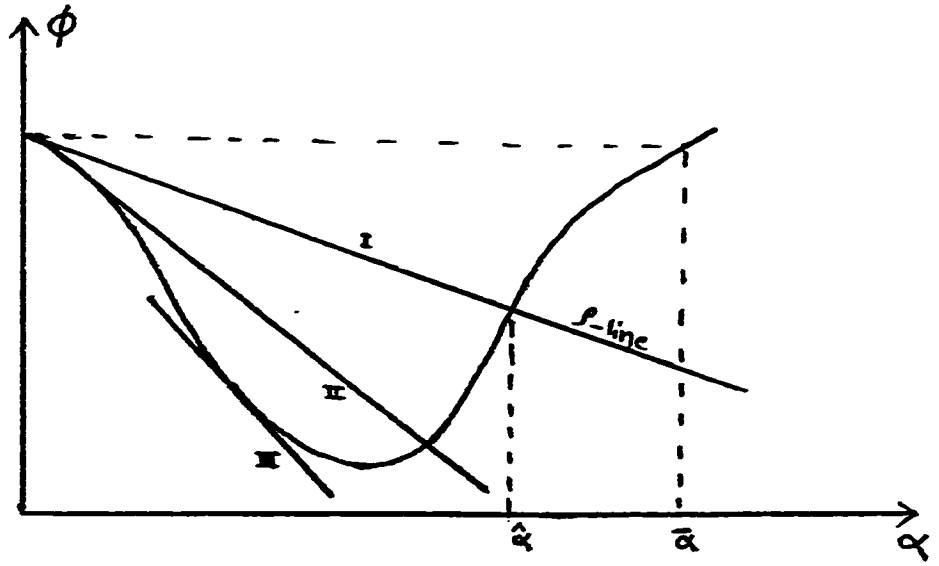
Remark The requirement that $\rho < \frac{1}{2}$ allows the property that the minimizing value of a quadratic function is acceptable . However, when $\phi(\alpha)$ is non-quadratic , the second condition (II) may exclude the minimizing point of $\phi(\alpha)$ (see the figure).

Wolfe conditions Because of the reason given in the above remark, Wolfe ([17],1968) replaced Goldestein's condition (II) by the following test for the slopes of tangent lines to $\phi(\alpha)$:

III - $\phi'(\alpha) \geq \sigma\phi'(0)$, $\sigma \in (\rho, 1)$.

An advantage of this test is that it is invariant to scale-factor changes whereas (II) is not [9].

Now condition (I) is used to exclude the right-hand extreme while (III)



is used to exclude the left-hand extreme of the interval $[0, \bar{\alpha}]$.

The restriction that $\sigma > \rho$ ensures that acceptable points (the points satisfying (I) and (III)) exist and can be located in a finite number of steps. On the other hand , if $\sigma < \rho$ Al-Baali and Fletcher [1] constructed the following C^1 function having no acceptable points:

Example [1] Consider the C^1 quadratic spline:

$$\phi(\alpha) = \begin{cases} (1/2)[(1-\sigma^+)/t]\alpha^2 - \alpha & \text{if } \alpha \le t \\ (1/2)(\sigma^+ - 1)t - \sigma^+\alpha & \text{if } \alpha \ge t \end{cases}$$

where σ^+ is marginally greater than σ , and $t \in (0,1)$. Note that $\phi'(0) = -1$. If we choose

$$t = 2(\rho - \sigma^+)/(1 - \sigma^+),$$

then the graph of $\phi(\alpha)$ itersects the ρ-line at $\bar{\alpha} = 1$. Because $\rho < 1/2$, we

have $t \in (0,1)$ as required . Finally, $\phi'(\alpha) = -\sigma^+$ for $\alpha \geq t$, and $\phi(\alpha) < -\sigma^+$ for $\alpha < t$, so it follows that (IV) is not satisfied for any $\alpha \in [0,1]$.

In fact, in some cases when $\phi \longrightarrow -\infty$, acceptable points may not exist and line search algorithm will continue to reduce ϕ indefinitely. Suppose that this is not the case, then the graph of ϕ must intersect the $\rho - line$ defined by taking equality in I (see the fig. at p.17).

Let $\hat{\alpha}$ be the least value of $\alpha > 0$ at which this intersection occurs. Then the following lemma holds which proves the existence of acceptable points for conditions (I) and (III).

Lemma 2.1 [4] *If $\hat{\alpha}$ exists and $\sigma \geq \rho$ then there exists an interval of acceptable points satisfying (I),(III) in $(0,\hat{\alpha}]$ provided that ϕ is a continuously differentiable function.*

Proof We choose any point $\epsilon \in (0,\hat{\alpha})$ and apply the definition of $\hat{\alpha}$ to have

$$\phi(\epsilon) \leq \phi(0) + \epsilon\rho\phi'(0) \qquad (a)$$

and

$$\phi(\hat{\alpha}) = \phi(0) + \hat{\alpha}\rho\phi'(0) \qquad (b)$$

Applying the mean-value theorem we obtain $\alpha \in (\epsilon, \hat{\alpha})$ which satisfies (I). Now we arrive at

$$\phi'(\alpha) = \frac{\phi(\hat{\alpha}) - \phi(\epsilon)}{\hat{\alpha} - \epsilon} > \rho\phi'(0) \geq \sigma\phi'(0)$$

after using the relations (a)-(b) and also the facts $\sigma > \rho$ and $\phi'(0) < 0$. This shows that α satisfies both (I) and (III) strictly and hence an interval of acceptable points exists by continuity of ϕ and ϕ' .□

Remark As pointed out by Fletcher [4], a more stringent two-sided test on the slope is preferred in place of (III) . It is as follows:

IV- $| \phi'(\alpha) | \leq -\sigma\phi'(0)$

which can be written as

$$| g_{k+1}^T d_k | \leq -\sigma g_k^T d_k.$$

Notice that an acceptable point in (I) and (III) may not be close to the exact minimizer of ϕ , whereas by making σ small we can find an acceptable point in I and IV which is arbitrarily close to a local exact minimizer of ϕ. In practice, values of $\sigma = 0.1$ and $\rho = 0.01$ have produced encouraging results [1]. The following lemma guarantees the existence of acceptable points when the two-sided test is used in place of III.

Lemma 2.2 [4] *If $\hat{\alpha}$ exists and $\sigma > \rho$ then there exists an interval of points satisfying (I) , (IV) in $(0, \hat{\alpha}]$ provided that ϕ is continuously differentiable.*

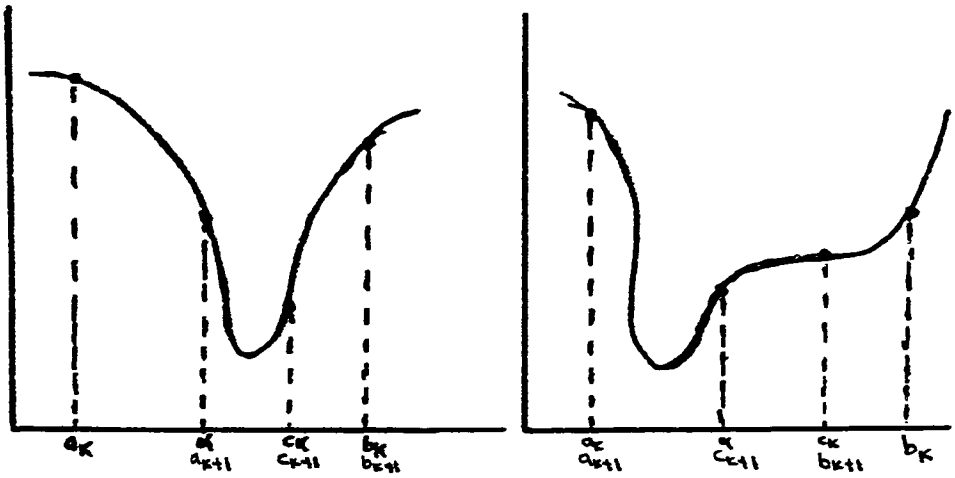
Proof: With the same notation as described in lemma 2.1 , by making ϵ sufficiently small, we can ensure that $\phi(0) > \phi(\epsilon) > \phi(\hat{\alpha})$. It then follows that the point α which exists by virtue of the mean-value theorem satisfies $0 > \phi'(\alpha) > \sigma\phi'(0)$. Thus α is strictly acceptable for I and IV. The lemma now follows by the continuity of ϕ and ϕ'. □

To avoid the possibility that $\phi(\alpha) \longrightarrow -\infty$ without having an acceptable point , we assume that we are able to supply an estimate for a lower bound $\overline{\phi}$ of $\phi(\alpha)$ for $\alpha > 0$. More precisely , we assume that we are prepared to accept any value of α for which $\phi(\alpha) < \overline{\phi}$, where $\overline{\phi} < \phi(0)$.[1]

2.4 LINE SEARCH WITH CONDITONS (I) AND (IV)

Since quadratic fit or interpolation is the most useful scheme in line searching, we will give here an algorithm using the idea of quadratic fit, which was described in section 2.2.3, together with the conditions (I) and (IV) of the previous section .

After finding new point where the derivative of the quadratic fit q van-



ishes, the objective function f is evaluated at this point. Now one of the outer points must be replaced by the new point to begin the next iteration in such a way that the three new points are distributed over an interval bracketing a local minimum of f along the line. It may be noted that the point to be discarded need not be the one with higher functional value [14]. This helps us to fit a quadratic polynomial for f as we can see in the above figure. The algorithm which handles this situation is similar to the one given in [14].

In the following lines we have modified the algorithm so that it terminates when a point satisfying conditions (I) and (IV) is located.

Modified Quadratic Interpolation

Step 1. input a_1, b_1, c_1 where $a_1 < c_1 < b_1$

Step 2. set $\phi_a = \phi(a_1), \phi_b = \phi(b_1), \phi_c = \phi(c_1)$

Step 3. for $k = 1, 2, \ldots$ repeat

Step 4. set

$$\alpha_k = (1/2)\frac{(b_k^2 - c_k^2)\phi_a + (c_k^2 - a_k^2)\phi_b + (a_k^2 - b_k^2)\phi_c}{(b_k - c_k)\phi_a + (c_k - a_k)\phi_b + (a_k - b_k)\phi_c}$$

Step 5. $\phi_{\alpha_k} = \phi(\alpha_k)$

Step 6. If $\phi_{\alpha_k} \leq \phi(0) + \alpha_k \rho \phi'(0)$ and $\mid \phi'(\alpha_k) \mid \leq -\sigma\phi'(0)$

then return to main program.

Step 7. if $\alpha_k < c_k$ and $\phi_{\alpha_k} < \phi_c$ then

$$a_{k+1} = a_k, \quad b_{k+1} = c_k, \quad c_{k+1} = \alpha_k, \quad \phi_b = \phi_c, \quad \phi_c = \phi_{\alpha_k}$$

else if $\alpha_k > c_k$ and $\phi_{\alpha_k} > \phi_c$ then

$$a_{k+1} = a_k, \quad b_{k+1} = \alpha_k, \quad c_{k+1} = c_k, \quad \phi_b = \phi_{\alpha_k}$$

else if $\alpha_k < c_k$ and $\phi_{\alpha_k} > \phi_c$ then

$$a_{k+1} = \alpha_k, \quad b_{k+1} = b_k, \quad c_{k+1} = c_k, \quad \phi_a = \phi_{\alpha_k}$$

else set

$$a_{k+1} = c_k, \quad b_{k+1} = b_k, \quad c_{k+1} = \alpha_k, \quad \phi_a = \phi_c, \quad \phi_c = \phi_{\alpha_k}$$

Remark It may be noted that step (6) in the above algorithm provides us with a criterion to stop inexact line search.

Remark We have applied the Modified Quadratic Interpolation Algorithm for inexact line search to various test functions in order to obtain computational results in the last chapter.

CHAPTER 3

CONJUGATE GRADIENT METHODS

3.1 INTRODUCTION

The conjugate direction algorithms were introduced originally by Hestens and Stiefel in 1952 [7] who emphasized their application to the solution of system of linear equations.

Modifications and extensions of these methods have been made by numerous authors, the most notable are due to Davidon, Fletcher, and Powell. We can look at the conjugate gradient methods as being somewhat intermediate between the method of steepest descent and Newton's method. The conjugate gradient methods accelerate the slow convergence associated with steepest descent while avoiding the information requirements associated with the evaluation, storge, and inversion of the Hessian as required by Newton's method.

In this chapter we shall give a complete description of the conjugate gradient method and some of its modifications and extensions.

3.2 THE PURE CONJUGATE GRADIENT METHOD

Let Q be an $n \times n$ symmetric matrix. A function f expressible in the form

$$f(x) = \frac{1}{2} x^T Q x - b^T x \qquad , x \in \Re^n$$

is called a *quadratic function*. It will be convenient to say that f is positive definite when Q is positive definite.

Since a sufficiently smooth function behaves like a quadratic function near its local extrema, positive definite quadratic functions play a significant role in the devlopment of algorithms to solve unconstrained optimization problems. For this reason the conjugate direction methods were invented and analyzed for the purely positive quadratic functions. The techniques once worked out for these functions are then extended by approximation to more general problems.

In order to define conjugate direction method precisely, we need the following concepts:

Definition 3.1 [9] Given a symmetric $n \times n$ matrix Q , two distinct nonzero n-vectors d_1 and d_2 are said to be *Q-orthogonal* , or *conjugate* with respect to Q, if $d_1^T Q d_2 = 0$.

A finite set of n-vectors $d_0, d_1, ..., d_k$ is said to be Q-orthogonal set if $d_i^T Q d_j = 0 \quad i \neq j \quad , i, j = 0, 1, \ldots, k.$

Similarly , we say that two straight lines in $\Re^n$ are mutually conjugate with respect to Q if their direction vectors are Q-orthogonal.

Propostion 3.1 [9] *If Q is positive definite and the set of nonzero vectors $d_0, d_1, ..., d_k$ are Q-orthogonal then these vectors are linearly independent.*

Proof Suppose there are constants $\alpha_i, i = 0, 1, ..., k$ such that

$$\alpha_0 d_0 + \cdots + \alpha_k d_k = 0.$$

Multiplying both sides by Q we get

$$\alpha_0 Q d_0 + \cdots + \alpha_k Q d_k = 0.$$

Taking the scalar product with d_i yields

$$\alpha_0 d_i^T Q d_0 + \cdots + \alpha_i d_i^T Q d_i + \cdots + \alpha_k d_i^T Q d_k = 0.$$

Since the vectors $d_0, \ldots, d_k$ are Q-orthogonal , $d_i^T Q d_j = 0$ if $i \neq j$. Thus we have $\alpha_i d_i^T Q d_i = 0$. Since $d_i^T Q d_i > 0$, in view of the positive definitness of Q, we have $\alpha_i = 0$. □

Remark From the above proposition, it is easy to see that the number of mutually Q-orthogonal vevtors in $\Re^n$ is exactly n.

Now , consider a positive definite quadratic function

$$f(x) = \frac{1}{2}x^T Qx - b^T x \qquad , x \in \Re^n$$

which is to be minimized. If we minimize f successively along n straight lines $L_1, \ldots L_n$ which are mutually conjugate with respect to Q, then this procedure is called a *conjugate directions method.*

The Conjugate Gradient method is a particular form of Conjugate direction method where the line L_1 is chosen through an arbitrary point $x_1 \in \Re^n$ in the direction of the steepest descent vector

$$d_1 = -\nabla f(x_1) = -g_1.$$

Then at each iteration k the direction d_k of the line L_k is obtained by combining linearly the gradient g_k of the function f at x_k (Here x_k is the point where f attains its minimum along the line L_{k-1}) and the previous directions $d_1, d_2, \ldots, d_{k-1}$. The coefficients of the linear combination are chosen in such a way that d_k is conjugate to all preceding directions $d_1, \ldots, d_{k-1}$. We shall refer to conjugate gradient method as C-G method in our further discussion.

In the following lines we describe a procedure for the C-G method [7]:

Step I. Select an initial point x_1, and compute the steepest descent vector $d_1 = -\nabla f(x_1) = -g_1$.

Step II. We consider the line L_1 through x_1 in the direction of d_1, and find $x_2 \in L_1$ where f attains its minimum value on L_1 . (Note that the (n-1)-plane through x_2 conjugate to $d_1$contains the minimum point x^* of f. Thus the dimension for our space of search is diminished by one).

Step III. We repeat the same process on the (n-1)-plane ,that is, we select a steepest descent vector d_2 of f at x_2 in the (n-1)-plane and then find the point x_3 where f attains its minimum on the line L_2 through x_2 in the direction d_2.

Again the (n-2)-plane through x_3 and conjugate to d_2 contains x^*, so we repeat the process on this plane.

Remark We observe that the dimension of search space decreases by one at each step. Therefore, after m steps where $m \leq n$, we obtain a point x_{m+1} which coincides with the minimum point x^* of f.

Fortunately, in applications we do not require a complete description of

the planes involved in the foregoing working rule. All what we need is a formula to compute the directions $d_i's$ of the conjugate lines where f is to be minimized and another formula for the step sizes $\alpha_i's$.

In the rest of this section we justify the explicit representations of $\alpha_i's$ and $d_i's$ which are given as follows [9]:

$$\alpha_k = -\frac{g_k^T d_k}{d_k^T Q d_k} \tag{1}$$

$$d_{k+1} = -g_{k+1} + \beta_k d_k \tag{2}$$

where

$$\beta_k = \frac{g_{k+1}^T Q d_k}{d_k^T Q d_k} \tag{3}$$

To establish these formulas , consider the quadratic problem:

$$\min(\frac{1}{2}x^T Q x - b^T x) \tag{4}$$

where Q is positive definite . We know that the unique solution to this problem is also the unique solution to the linear equation

$$Qx = b. \tag{5}$$

Let $d_0, d_1, \ldots, d_{n-1}$ be n nonzero Q-orthogonal vectors . From propsition 2.1 we know that these vectors are linearly independent . Suppose that x^* is

the solution of (4) or (5) . Now , from the previous description of the C-G method , suppose we start with $x_0 \in \Re^n$. Then at each step k , the point x_{k+1} where f attains its minimum on the line L_k is generated according to the schemes

$$x_{k+1} = x_k + \alpha_k d_k \qquad , k \geq 0. \tag{6}$$

Here α_k is known as *stepsize* on the line L_k. Since the $d_k's$ are linearly independent we can write

$$x^* - x_0 = \alpha_0 d_0 + \cdots + \alpha_{n-1} d_{n-1}$$

for certain constants $\alpha_k's$. If we multiply the above equation by Q and take the scalar product with d_k we get

$$\alpha_k = \frac{d_k^T Q(x^* - x_0)}{d_k^T Q d_k}. \tag{7}$$

Now following the iterative process (6) we have

$$x_1 = x_0 + \alpha_0 d_0$$

$$x_2 = x_1 + \alpha_1 d_1 = x_0 + \alpha_0 d_0 + \alpha_1 d_1$$

$$\vdots$$

$$x_k = x_0 + \alpha_0 d_0 + \cdots + \alpha_{k-1} d_{k-1}.$$

Therefore ,

$$x_k - x_0 = \alpha_0 d_0 + \cdots + \alpha_{k-1} d_{k-1}$$

Now multiplying by Q and taking the scalar product with d_k we get (from the Q-orthogonality of $d_k's$)

$$d_k^T Q(x_k - x_0) = 0.$$

From (7) we can write α_k as

$$\alpha_k = \frac{d_k^T Q(x^* - x_k + x_k - x_0)}{d_k^T Q d_k}$$

$$= \frac{d_k^T Q(x^* - x_k) + d_k^T Q(x_k - x_0)}{d_k^T Q d_k}.$$

Therefore,

$$\alpha_k = \frac{d_k^T Q(x^* - x_k)}{d_k^T Q d_k}$$

$$= \frac{d_k^T (Qx^* - Qx_k)}{d_k^T Q d_k}$$

$$= \frac{d_k^T (b - Qx_k)}{d_k^T Q d_k}$$

Since $g_k = Qx_k - b$, we get

$$\alpha_k = -\frac{g_k^T d_k}{d_k^T Q d_k}$$

which is (1).

Before justifying (2) and (3) it is important to note that

$$g_k^T d_i = 0, \qquad i < k. \tag{8}$$

To show this , first note that

$$x_{k+1} - x_k = \alpha_k d_k$$

and

$$g_{k+1} - g_k = Q(x_{k+1} - x_k) = \alpha_k Q d_k$$

We will show (8) by induction :

For $k = 1$ we have

$$g_1 = g_0 + \alpha_0 Q d_0.$$

Multiplying by d_0^T we get

$$d_0^T g_1 = d_0^T g_0 + \alpha_0 d_0^T Q d_0.$$

Substituting the value of α_k from (1) we get

$$d_0^T g_1 = 0$$

Assume that (8) is true for k, i.e., $d_i^T g_k = 0$ for $i < k$. Now we show that (8) holds for $k + 1$. Note that

$$g_{k+1} = g_k + \alpha_k Q d_k.$$

Multiplying the above equation by d_i, we get

$$d_i^T g_{k+1} = d_i^T g_k + \alpha_k d_i^T Q d_k.$$

The first term on the right-hand side vanishes because of the induction hypothesis while the second one vanishes by the Q-orthogonality of the d_is. Thus $g_k^T d_i = 0$ for $i < k+1$

Now, instead of specifying the directions at hand, we can determine them sequentially at each step of the iteration. At step k we evaluate the current negative gradient vector and add to it a linear combination of the previous direction vectors to obtain a new conjugate direction vector along which to move, i.e., $d_{k+1} = -g_{k+1} + \sum_{j=0}^{k} \beta_j d_j$ or

$$d_{k+1} = -g_{k+1} + \beta_1 d_1 + \cdots + \beta_k d_k. \tag{9}$$

Now multiplying by Q and taking the scalar product with d_k we get

$$d_k^T Q d_{k+1} = -d_k^T Q g_{k+1} + \cdots + \beta_k d_k^T Q d_k.$$

Thus, by Q-conjugacy of directions, we have

$$0 = -d_k^T Q g_{k+1} + 0 + \cdots + 0 + \beta_k d_k^T Q d_k,$$

i.e.

$$\beta_k = \frac{g_{k+1}^T Q d_k}{d_k^T Q d_k}$$

which is (3).

In fact, for quadratic functions, we have $\beta_i d_i = 0$ for all $i < k$. This is seen by using (8). Thus the expression (9) simplifies to $d_{k+1} = -g_{k+1} + \beta_k d_k$ which is (2) □

Remark An alternative formula for α_k which is helpful in application is as follows :

$$\alpha_k = \frac{g_k^T g_k}{d_k^T Q d_k}.$$

This is because

$$-g_k^T d_k = -g_k^T(-g_k + \beta_{k-1} d_{k-1})$$

$$= g_k^T g_k - \beta_{k-1} g_k^T d_{k-1}$$

and by using (8) we get

$$-g_k^T d_k = g_k^T g_k.$$

So α_k can be written as

$$\alpha_k = \frac{g_k^T g_k}{d_k^T Q d_k}.$$

Also we note that

$$g_k^T d_k = - \| g_k \|^2$$

since $-g_k^T d_k = g_k^T g_k$. Therefore, we have

$$g_k^T d_k < 0 \tag{10}$$

which is an important fact known as *descent property.*

Now we can formulate

Conjugate Gradient Algorithm (quadratic case)[9]

step 1. Starting at x_0 compute $g_0 = Qx_0 - b$ and set $d_0 = -g_0$

step 2. For $k = 0, 1, \ldots, n$

a) compute $\alpha_k = \frac{-g_k^T d_k}{d_k^T Q d_k} = \frac{g_k^T g_k}{d_k^T Q d_k}$

and set $x_{k+1} = x_k + \alpha_k d_k$

b) compute $g_{k+1} = Qx_{k+1} - b$

c) compute $\beta_k = \frac{g_{k+1}^T Q d_k}{d_k^T Q d_k}$

and set $d_{k+1} = -g_{k+1} + \beta_k d_k$.

Termination: The algorithm terminates at the mth iteration if $g_{m+1} = 0$ for $m < n$ and $x_{m+1} = x^*$, the minimum point of f [9].

To see the performance of the algorithm we consider the following example:

Example [7] Let $f(x) = \frac{1}{2}x^T Q x - b^T x$

where $Q = \begin{bmatrix} 6 & 13 & -17 \\ 13 & 29 & -38 \\ -17 & -38 & 50 \end{bmatrix}$

and $b = (1, 2, -3)^T$. f achieves its minimum at $x^* = (1, -3, -2)^T$. We shall start with $(x1, x2, x3)^T = (1, 0, 0)^T$.

The result is illustrated in the following table where f reprsents the value of the function at the point $(x1, x2, x3)^T$ corresponding to various steps.

Iter.	x1	x2	x3	f
0	1	0	0	2
1	.9409796	-.129845	.1652573	-.0185
2	-.1096129	-1.465189	-1.210444	-.3594632
3	.9999646	-2.999928	-1.99998	-.5000088

It may be noted that the third iterate is fairly close to x^*, the minimum of the quadratic function f which depends on a 3×3 matrix Q.

3.3 QUADRATIC APPROXIMATION

There are a number of ways to extend the conjugate gradient method to nonquadratic problems by making suitable approximations to the algorithm. One of these ways is the *quadratic approximation.* [8]

In the *quadratic approximation method* we approximate the objective function f by the following quadratic function q at each step k :

$$q(x) = f(x_k) + \nabla f(x_k)^T x + (1/2) x^T \nabla^2 f(x_k) x.$$

Here $\nabla^2 f(x_k) = F(x_k)$ is the Hessian of f at x_k. Now we apply the C-G method to the quadratic function q where we have

$$g_k = \nabla f(x_k) \quad , \quad Q = \nabla^2 f(x_k) = F(x_k).$$

With the above approximation, we can not guarantee the termination of the method within n steps. It is then possible to continue finding new directions according to the algorithm and stop when some termination criterion is achieved. Otherwise, the conjugate gradient process is postponed after n or n+1 steps and we reset d_k to the steepest descent direction [10]. This strategy is useful in the following sense [11]:

If the iterates progress from a nonquadratic region into a neighbourhood of the solution in which $f(x)$ is closely approximated by a quadratic, then

the reset method can be expected to converge rapidly. It may be noted that the nonreset method may not converge in this situation.

The general conjugate gradient algorithm is then defined as follows [9]:

step 1 . Starting at x_0 compute $g_0 = \nabla f(x_0)^T$ and set $d_0 = -g_0$

step 2 . For $k = 0, 1, \ldots, n-1$

a) set $x_{k+1} = x_k + \alpha_k d_k$ where $\alpha_k = \frac{-g_k^T d_k}{d_k^T F(x_k) d_k}$

b) comput $g_{k+1} = \nabla f(x_{k+1})^T$

c) unless $k = n-1$, set $d_{k+1} = -g_{k+1} + \beta_k d_k$ where $\beta_k = \frac{g_{k+1}^T F(x_k) d_k}{d_k^T F(x_k) d_k}$

and repeat (a)

step 3 . Replace x_0 by x_n and go back to step 1 .

A major disadvantage of this algorithm is the evaluation of $F(x_k)$ at each point, which is often computationaly expensive. In addition to this drawback, the algorithm is not globally convergent. These problems may be avoided if we do not consider $F(x_k)$ to be the replacement of the matrix Q in the formula (3). This argument forces us to determine the stepsize α_k by a line search that minimizes the objective function while ignoring the formula (1). Thus the new direction d_{k+1} (cf. (2)) must be found with an alternative

formula for β_k. In either case, the new values of α and β must agree with the old one in the quadratic case. This leads us to Fletcher-Reeves method and Polak-Ribiere method.

3.4 FLETCHER-REEVES METHOD

The basis for the conjugate gradient method due to Fletcher and Reeves (1964) was provided directly from the conjugate gradient method of Hestenes and Stiefel (1952) [4].

Fletcher-Reeves method uses a line search in order to find a suitable stepsize α_k while minimizing f along a line. Also, in this method, the formula for β_k given in (3) is replaced by the following formula:

$$\beta_k = \frac{g_{k+1}^T g_{k+1}}{g_k^T g_k}. \tag{11}$$

To justify this, first we note that $d_k = -g_k + \beta_{k-1} d_{k-1}$ which on multiplying by g_{k+1}^T gives us

$$g_{k+1}^T d_k = -g_{k+1}^T g_k + \beta_{k-1} g_{k+1}^T d_{k-1}.$$

Using (8) in the above equation we obtain

$$g_{k+1}^T g_k = 0. \tag{12}$$

Since

$$\alpha_k Q d_k = g_{k+1} - g_k,$$

we have

$$Q d_k = \frac{1}{\alpha_k}(g_{k+1} - g_k).$$

Multipling by g_{k+1} and using (12) we get

$$g_{k+1}^T Q d_k = \frac{1}{\alpha_k} g_{k+1}^T g_{k+1}.$$

The above equation together with (3) leads us to

$$\beta_k = \frac{\frac{1}{\alpha_k} g_{k+1}^T g_{k+1}}{d_k^T Q d_k}.$$

From the defintion of α_k we derive

$$\beta_k = \frac{g_{k+1}^T g_{k+1}}{g_k^T g_k} \qquad \square$$

The Fletcher-Reeves algorithm (using restart) can be described as follows [9]:

step 1. Given x_0 , compute $g_0 = \nabla f(x_0)^T$ and set $d_0 = -g_0$

step 2. For $k = 0, 1, \ldots, n-1$,

search that minimizes the objective function while ignoring the formula (1). Thus the new direction d_{k+1} (cf. (2)) must be found with an alternative formula for β_k. In either case, the new values of α and β must agree with the old one in the quadratic case. This leads us to Fletcher-Reeves method and Polak-Ribiere method.

3.4 FLETCHER-REEVES METHOD

The basis for the conjugate gradient method due to Fletcher and Reeves (1964) was provided directly from the conjugate gradient method of Hestenes and Stiefel (1952) [4].

Fletcher-Reeves method uses a line search in order to find a suitable stepsize α_k while minimizing f along a line. Also, in this method, the formula for β_k given in (3) is replaced by the following formula:

$$\beta_k = \frac{g_{k+1}^T g_{k+1}}{g_k^T g_k}. \tag{11}$$

To justify this, first we note that $d_k = -g_k + \beta_{k-1} d_{k-1}$ which on multiplying by g_{k+1}^T gives us

$$g_{k+1}^T d_k = -g_{k+1}^T g_k + \beta_{k-1} g_{k+1}^T d_{k-1}.$$

Thus

$$\| d_k \|^2 = \sum_{i=1}^{k} (\Pi_{j=i}^{k-1} \beta_j^2) \| g_i \|^2$$

which on using the definition of β_k gives us

$$\| d_k \|^2 = \sum_{i=1}^{k} \frac{\| g_k \|^4}{\| g_i \|^2}. \tag{14}$$

Now, if we assume that (13) is not true then there exists a constant ϵ such that $\| g_k \| \geq \epsilon$ for all k, and because $\| g_k \|$ is also bounded above on the level set l, we have

$$\| d_k \|^2 \leq kc$$

where c is a positive constant.

Consider the directional derivatives:

$$s_k = \frac{d_k^T g_k}{\| d_k \| \| g_k \|}. \tag{15}$$

From the definition of α_k and d_k, (15) can be written as

$$s_k = -\frac{\| g_k \|}{\| d_k \|}.$$

Thus the series $\sum_k s_k^2$ is divergent. From the quadratic approximation of f at $x \in l$ we have

$$f(x) = f(x_k) + (x - x_k)^T g_k + (1/2)(x - x_k)^T \nabla^2 f(x_k)(x - x_k).$$

If λ is an upper bound of $\| \nabla^2 f(x) \|$, we obtain

$$f(x) \leq f(x_k) + (x - x_k)^T g_k + (1/2)\lambda \| x - x_k \|^2$$

which after some considerations of rates of change of first derivatives , gives the inequality

$$f(x_{k+1}) \leq f(x_k) - s_k^2 \| g_k \|^2 / (2\lambda).$$

Because $f(x)$ is bounded above on l, and for $x_k \in l$ we note that $\| g_k \|$ is bounded above, therefore the series

$$\sum s_k^2 \| g_k \|^2$$

is convergent. This contradicts the fact that $\sum_k s_k^2$ is divergent as established above. It therefore follows that our assumption is false, and the limit (13) must be achieved. □

Now we show that the descent property holds in all iterations and the limit (13) is achieved for the Fletcher-Reeves method whenever inexact line search is used for a twice contiuously differentiable function f.

First we have the following theorem:

Theorem 3.2 [2] *Let a step size α_k be determined by inexact line search in*

the Fletcher-Reeves method (cf. p. 37). If α_k satisfies the condition (IV) (cf. p. 18) with $\sigma \in (0, 1/2]$ and if the gradient vector $g_k = \nabla f(x_k)$ is nonzero, then the descent property holds, i.e.,

$$g_k^T d_k < 0.$$

Proof First we shall use induction to show that the inequality

$$-\sum_{j=0}^{k-1} \sigma^j \leq \frac{g_k^T d_k}{\| g_k \|^2} \leq -2 + \sum_{j=0}^{k-1} \sigma^j \tag{16}$$

holds for all k , where $g_k \neq 0$, then it will follow inductively that descent property holds. For $k = 1$ equation (16) is clearly satisfied .

Now , assume that (16) is true for $k \geq 1$. Since

$$\sum_{j=0}^{k-1} \sigma^j < \sum_{j=0}^{\infty} \sigma^j = \frac{1}{1-\sigma} \tag{17}$$

and $\sigma \in (0, \frac{1}{2}]$ therefore $1/(1-\sigma) \leq 2$. This shows that the right-hand side of (16) is also negative. Hence $g_k^T d_k < 0$,i.e., the descent property is satisfied on iteration k .

From the definition of d_{k+1} and β_k in the Fletcher-Reeves method, we have

$$d_{k+1} = -g_{k+1} + \frac{\| g_{k+1} \|^2}{\| g_k \|^2} d_k.$$

Multiplying by $g_{k+1}^T / \| g_{k+1} \|^2$, we get

$$\frac{g_{k+1}^T d_{k+1}}{\| g_{k+1} \|^2} = -1 + \frac{g_{k+1}^T d_k}{\| g_k \|^2}. \tag{18}$$

From condition (IV) we have

$$| g_{k+1}^T d_k | \leq -\sigma g_k^T d_k.$$

Therfore ,

$$\sigma g_k^T d_k \leq g_{k+1}^T d_k \leq -\sigma g_k^T d_k.$$

Dividing by $\| g_k \|^2$ we get

$$\sigma \frac{g_k^T d_k}{\| g_k \|^2} \leq \frac{g_{k+1}^T d_k}{\| g_k \|^2} \leq -\sigma \frac{g_k^T d_k}{\| g_k \|^2}.$$

Adding (-1) to all sides leads us to

$$-1 + \sigma \frac{g_k^T d_k}{\| g_k \|^2} \leq -1 + \frac{g_{k+1}^T d_k}{\| g_k \|^2} \leq -1 - \sigma \frac{g_k^T d_k}{\| g_k \|^2}.$$

Using (18) in the above inequality we have

$$-1 + \sigma \frac{g_k^T d_k}{\| g_k \|^2} \leq \frac{g_{k+1}^T d_{k+1}}{\| g_{k+1} \|^2} \leq -1 - \sigma \frac{g_k^T d_k}{\| g_k \|^2}.$$

Now from (16) we obtain

$$-1 - \sigma \sum_{j=0}^{k-1} \sigma^j \leq \frac{g_{k+1}^T d_{k+1}}{\| g_{k+1} \|^2} \leq -1 + \sigma \sum_{j=0}^{k-1} \sigma^j. \tag{19}$$

The left-hand side of this inequality is

$$-1-\sigma\sum_{j=0}^{k-1}\sigma^j=-\sum_{j=0}^{k}\sigma^j$$

and the right-hand side is

$$-1+\sigma\sum_{j=0}^{k-1}\sigma^j=-2+\sum_{j=0}^{k}\sigma^j.$$

Thus (19) is equal to

$$-\sum_{j=0}^{k}\sigma^j\leq\frac{g_{k+1}^T d_{k+1}}{\|g_{k+1}\|^2}\leq -2+\sum_{j=0}^{k}\sigma^2$$

which is the same as (16) with k replaced by k+1 .Thus by induction hypothesis, the theorem is established. □

A consequence of the descent property is the following global convergance result:

Theorem 3.3 [2] *If the set l given in theorem 3.1 is bounded, if $f(x)$ is twice continuously differentiable and if α_k is any value satisfying the conditions (I) and (IV) (cf. 2.3) with $\rho<\sigma<\frac{1}{2}$, then the limit (13) is achieved for the Fletcher-Reeves method.*

Proof It is shown in theorem 3.2 that the descent property holds for $\sigma\in$

$(0, \frac{1}{2}]$. Therefore from condition (IV) and equations (16) - (17) it follows that

$$| g_k^T d_{k-1} | \leq -\sigma g_{k-1}^T d_{k-1} \leq \frac{\sigma}{1-\sigma} \| g_{k-1} \|^2 . \tag{20}$$

From the definition of d_k we have

$$\| d_k \|^2 = \| g_k \|^2 - 2\beta_{k-1} g_k^T d_{k-1} + \beta_{k-1}^2 \| d_{k-1} \|^2 .$$

Now the definition of β_k given in the Fletcher-Reeves method and the inequality (20) lead us to

$$\| d_k \|^2 \leq \| g_k \|^2 + \frac{2\sigma}{1-\sigma} \| g_k \|^2 + \beta_{k-1}^2 \| d_{k-1} \|^2$$

$$\leq (\frac{1+\sigma}{1-\sigma}) \| g_k \|^2 + \beta_{k-1}^2 \| d_{k-1} \|^2 .$$

It follows by induction that

$$\| d_k \|^2 \leq (\frac{1+\sigma}{1-\sigma}) \| g_k \|^4 \sum_{j=0}^{k} \| g_j \|^{-2} . \tag{21}$$

Now suppose that (13) is NOT true. Then there exists a constant , say ϵ , such that $\| g_k \| \geq \epsilon > 0$ for all k. Also g_k is bounded above on the level set l. Thus, from (21) we get

$$\| d_k \|^2 \leq c_1 k, \tag{22}$$

where c_1 is a positive constant.

From (16) and (17)

$$\frac{-g_k^T d_k}{\| g_k \| \| d_k \|} \geq (\frac{1-2\sigma}{1-\sigma}) \frac{\| g_k \|}{\| d_k \|}.$$

Let θ_k be the angle between d_k and $(-g_k)$, then we will have

$$\cos\theta_k = \frac{-g_k^T d_k}{\| g_k \| \| d_k \|}.$$

It follows that

$$\cos\theta_k \geq (\frac{1-2\sigma}{1-\sigma}) \frac{\| g_k \|}{\| d_k \|}. \tag{23}$$

Because $\sigma < 1/2$, we obtain from (22) and (23) that

$$\sum_k \cos^2\theta_k \geq (\frac{1-2\sigma}{1-\sigma})^2 \sum_k \| g_k \|^2 / \| d_k \|^2 \geq c_2 \sum_k k^{-1} \tag{24}$$

where c_2 is a positive constant. Hence the series $\sum_k \cos^2\theta_k$ diverges.

If λ is an upper bound on $\| \nabla^2 f(x) \|$ where $x \in l$ then we have

$$g_{k+1}^T d_k \leq g_k^T d_k + \lambda \alpha_k \| d_k \|^2 .$$

Thus by using condition (IV) we obtain

$$\sigma g_k^T d_k \leq g_k^T d_k + \lambda \alpha_k \| d_k \|^2,$$

therefore

$$\alpha_k \geq -\frac{(1-\sigma)}{\lambda \| d_k \|^2} g_k^T d_k$$

which can be substituted into condition (I) to give

$$f_{k+1} \leq f_k - \rho \frac{(1-\sigma)}{\lambda \parallel d_k \parallel^2} (g_k^T d_k)^2.$$

Upon using the definition of $\cos\theta$ it follows that

$$f_{k+1} \leq f_k - c_3 \parallel g_k \parallel^2 \cos^2 \theta_k$$

where $c_3 = \rho(1-\sigma)/\lambda > 0$. Since $f(x)$ is bounded, $\sum_k \parallel g_k \parallel^2 \cos^2 \theta_k$ is convergent . Because $\parallel g_k \parallel$ is bounded below this contardicts (24). Hence, our assumption that (13) does not hold is false. □

3.5 POLAK-RIBIERE METHOD

Another important method which uses the same idea as applied in the Fletcher-Reeves method is due to Polak and Ribiere (1969) where an alternative formula for β_k is used [9]:

$$\beta_k = \frac{g_{k+1}^T (g_{k+1} - g_k)}{g_k^T g_k} \tag{25}$$

The development of this formula is as follows:

Since $\alpha_k Q d_k = g_{k+1} - g_k$, taking the scalar product with g_{k+1} we get

$$\alpha_k g_{k+1}^T Q d_k = g_{k+1}^T (g_{k+1} - g_k).$$

Therefore,

$$g_{k+1}^T Q d_k = \frac{1}{\alpha_k} g_{k+1}^T (g_{k+1} - g_k).$$

Substituting this in (3) we obtain

$$\beta_k = \frac{\frac{1}{\alpha_k} g_{k+1}^T (g_{k+1} - g_k)}{d_k^T Q d_k}.$$

From the definition of α_k we get

$$\beta_k = \frac{g_{k+1}^T (g_{k+1} - g_k)}{g_k^T g_k}.$$

Thus the algorithm of Polak-Ribiere method is the same as described for the Fletcher-Reeves method except that (25) replaces (11) in Polak-Ribiere method. As pointed out earlier, both algorithms due to Fletcher-Reeves and Polak-Ribiere are equivalent to the conjugate gradient algorithm in the quadratic case.

Unlike the Fletcher-Reeves method, it has not been possible to establish the global convergance results for the Polak-Ribiere method unless we assume that the step lengths $\| x_{k+1} - x_k \|$ tend to zero .

On the other hand , in numerical computations the Polak-Ribiere formula for β_k is generally far more successful than the Fletcher-Reeves formula. For the justification we have the following remark .

Remark [12] In several numerical calculations, formula (25) is far more successful than formula (11).

To explain this remark we compare the change $\| x_{k+1} - x_k \|$ made by the C-G method for two consecutive iterations . Suppose that at the k-th iteration, the C-G method has made a very small change in $\| x_{k+1} - x_k \|$. We ask whether the next iteration can be equally bad. A relatively tiny value of $\| x_{k+1} - x_k \|$ occurs only if the angle θ_k between d_k and $(-g_k)$ is close to $\pi/2$. Therefore, from the definition of $\cos\theta_k, \alpha_k$ and β_k

$$\cos\theta_k = \frac{\| g_k \|}{\| d_k \|}.$$

Thus, we must have $\| d_k \| >> \| g_k \|$, while avoiding $\| d_{k+1} \| >> \| g_{k+1} \|$. Now, the relatively small value of $\| x_{k+1} - x_k \|$ gives $g_{k+1} \approx g_k$. Thus the Fletcher-Reeves formula yields $\beta_{k+1} \approx 1$, while the Polak-Ribiere formula yields $| \beta_{k+1} | << 1$. Further , using the definition of d_{k+1} we deduce the relation

$$\| d_{k+1} \| = (\| g_{k+1} \|^2 + \beta_k^2 \| d_k \|^2)^{1/2}.$$

It follows that the Fletcher-Reeves formula gives $\| d_{k+1} \| \approx \| d_k \|$, but the Polak-Ribiere formula gives $\| d_{k+1} \| << \| d_k \|$ as required [13].

3.6 SHANNO METHOD

Rather than restarting the C-G method after n or n+1 steps, Shanno ([15],1985) implemented an angle test to determine when to restart conjugate gradient method in a steepest descent direction. The test guarantees that the cosine of the angle between the search direction d_i and the negative gradient $(-g_i)$ is within a constant multiple of the cosine of the angle between the Fletcher- Reeves search direction and the negative gradient [15].

Let θ_i be the angle between the search vector d_i and the negative gradient $(-g_i)$. It is well known that the C-G method will converge to a stationary point of $f(x)$ under very loose line search criteria if $\cos\theta_i$ is positive and bounded away from zero. We know that

$$\cos^2\theta_i = \frac{(g_i^T d_i)^2}{\| g_i \|^2 \| d_i \|^2}$$

and from the definition of α_i and β_i we get

$$\cos^2\theta_i = \frac{\| g_i \|^2}{\| d_i \|^2}.$$

Substituting the value of $\| d_k \|$ from (14) in the above equation, we obtain

$$\cos^2\theta_i = \frac{1}{\| g_i \|^2 \sum_{l=1}^{i} 1/\| g_l \|^2}.$$

As we have seen in theorem 3.1, a sufficient condition for the Fletcher-Reeves method to converge with an appropriately chosen α_i is that $\sum_{i=1}^{\infty} \cos^2 \theta_i$ diverges (see the proof of theorem 3.1 and theorem 3.3) which leads to the divergence of $\tau \sum_{i=0}^{\infty} \cos^2 \theta_i$ for any $\tau > 0$. From this observation, Shanno modified the conjugate gradient algorithm as follows:

step 1. For a given x_0, compute $g_0 = \nabla f(x_0)^T$ and set $d_0 = -g_0$.

step 2. For $k = 0, 1, \ldots$

a) set $x_{k+1} = x_k + \alpha_k d_k$ where α_k minimizes $f(x_k + \alpha d_k)$

b) compute $g_{k+1} = \nabla f(x_{k+1})^T$ then find

$$\gamma_{k+1}^2 = \frac{\tau}{\| g_{k+1} \|^2 \sum_{i=1}^{k+1} 1/\| g_i \|^2} \qquad , \tau > 0$$

c) compute $d_{k+1}^{\prime}$ by any desired C-G formula and let

$$\cos^2 \theta_{k+1} = \frac{(g_{k+1}^T d_{k+1}^{\prime})^2}{\| g_{k+1} \|^2 \| d_{k+1}^{\prime} \|^2}.$$

d) choose d_{k+1} as follows:

$$d_{k+1} = \begin{cases} d_{k+1}^{\prime} & \text{if } \cos^2 \theta_{k+1} \geq \gamma_{k+1}^2 \\ -g_{k+1} & \text{if } \cos^2 \theta_{k+1}^2 < \gamma_{k+1}^2 \end{cases}$$

Remark The above algorithm can easily be shown to converge to a stationary point. To see this, first we note that $\tau \leq 1$ gives us $\gamma_{k+1}^2 \leq 1$ (cf. (b)).

If $d_i = d_i^{\prime}$ then $\cos^2\theta_i \geq \gamma_i^2$. If $d_i = -g_i$ then $\cos^2\theta_i = 1$. Hence $\cos^2\theta_i \geq \gamma_i^2$ for all i and hence $\sum_{i=1}^{\infty}\cos^2\theta_i$ diverges due to the divergence of $\sum_{i=1}^{\infty}\gamma_i^2$.

3.7 TOUATI AHMED-STOREY METHOD

In order to get an advantage of computational aspects of the Polak- Ribiere formula for β_k and the useful theoretical features of Fletcher-Reeves formula, Touati Ahmed and Storey ([16],1990) gave a hybrid method based on both formulas. That is, the Polak-Ribiere formula is used whenever a certain conditon is satisfied. Otherwise we apply the Fletcher-Reeves formula.

Remark Touati Ahmed and Storey described three Hybrid algorithms. We shall confine ourselves only to the one they called Hybrid Algorithm 3, because it is globally convergent with respect to inexact line search whereas the first two are not.

Now we present the Hybrid Algorithm 3.

Hybrid Algorithm 3

Let $\lambda > 0$ and $(1/2) > \mu > \sigma$

step 1. For a given x_0,compute $g_0 = \nabla f(x_0)^T$ and set $d_0 = -g_0$

step 2. For $k = 0, 1, \ldots$

a) set $x_{k+1} = x_k + \alpha_k d_k$ where α_k minimizes $f(x_k + \alpha d_k)$

b) compute $g_{k+1} = \nabla f(x_{k+1})^T$

c) set $d_{k+1} = -g_{k+1} + \beta_k d_k$ where β_k is computed as follows: Let

$$\beta^{PR} = \text{Polak-Ribire formula (25)}$$

$$\beta^{FR} = \text{Fletcher-Reeves formula (11)}$$

i) If $\lambda \parallel g_{k+1} \parallel^2 \leq (2\mu)^{k+1}$ then go to (ii). Else $\beta_k = 0$ and continue.

ii) If $\beta_k^{PR} < 0$ then $\beta_k = \beta_k^{FR}$ and continue. Else go to (iii).

iii) If $\beta_k^{PR} \leq (\frac{1}{2\mu}) \parallel g_{k+1} \parallel^2 / \parallel g_k \parallel^2$ then $\beta_k = \beta_k^{PR}$ and continue.

Else $\beta_k = \beta_k^{FR}$ and continue. –

In the following theorem we show that Hybrid algorithm 3 has a descent property and is globally convergent when inexact line searches are performed to a twice continuously differentiable function f.

Theorem 3.4 [16] *If α_k is calculated which satisfies (IV) with $\sigma \in (0, 1/2)$ for all k such that $g_k \neq 0$ then the descent property (10) holds for Hybrid algorithm 3 for all such k.*

Proof The method of proof is the same as given for theorem 3.2 except that we replace σ by $\sigma/2\mu$. □

A consequence of this descent property is the following global convergence result.

Theorem 3.5 [16] *If the set l given in theorem 3.1 is bounded, if $f(x)$ is twice continuously differentiable, if α_k is any value satisfying the conditions (I) and (IV) (cf. 2.3) with $\rho < \sigma < 1/2$, and if β_k is computed as in Hybrid algorithm 3 at each iteration, then the limit (13) is achieved.*

Proof First note that in Hybrid algorithm 3 we ensure that

$$\lambda \parallel g_{k+1} \parallel^2 \leq (2\mu)^{k+1} \quad , \quad 1/2 > \mu > \sigma \qquad (26)$$

at each iteration. Also, all the β's computed by this algorithm satisfy

$$0 \leq \beta_{k-1} \leq (1/2\mu)[\parallel g_k \parallel^2 / \parallel g_{k-1} \parallel^2]. \qquad (27)$$

From the definition of d_k, we have

$$\parallel d_k \parallel^2 = \parallel g_k \parallel^2 - 2\beta_{k-1} g_k^T d_{k-1} + \beta_{k-1}^2 \parallel d_{k-1} \parallel^2 . \qquad (28)$$

It is shown in theorem 3.4 that the descent property holds for $\sigma \in (0, \frac{1}{2}]$. Thus from condition (IV) and theorem 3.4 it follows that

$$\mid g_k^T d_{k-1} \mid \leq -\sigma g_{k-1}^T d_{k-1} \leq \frac{\sigma}{1-(\sigma/2\mu)} \parallel g_{k-1} \parallel^2 \qquad (29)$$

which can be substituted into (28) to get

$$\parallel d_k \parallel^2 \leq \parallel g_k \parallel^2 + \frac{2\sigma}{1-(\sigma/2\mu)} \beta_{k-1} \parallel g_{k-1} \parallel^2 + \beta_{k-1}^2 \parallel d_{k-1} \parallel^2 .$$

Finally , using (27) , we obtain

$$\| d_k \|^2 \leq (\frac{1+(\sigma/2\mu)}{1-(\sigma/2\mu)}) \| g_k \|^2 + \beta_{k-1}^2 \| d_{k-1} \|^2 .$$

It follows by induction that

$$\| d_k \|^2 \leq (\frac{1+(\sigma/2\mu)}{1-(\sigma/2\mu)})(1/2\mu)^{2k} \| g_k \|^4 \sum_{l=1}^{k} \| g_l \|^{-2} .$$

Note that (26) holds for all iterations where a restart has not been made. For these iterations we therefore have

$$\| d_k \|^2 \leq (\frac{1+(\sigma/2\mu)}{1-(\sigma/2\mu)})(1/\lambda^2) \sum_{l=1}^{k} \| g_l \|^{-2} . \tag{30}$$

For those iterations where a restart has been made, we have

$$\| d_k \|^2 = \| g_k \|^2 . \tag{31}$$

Now suppose that (13) is NOT true. Then there exists a constant, say ϵ, such that $\| g_k \| \geq \epsilon > 0$ for all k. Hence from (30) we get

$$\| d_k \|^2 \leq c_1 k \tag{32}$$

where c_1 is a positive constant.

From theorem 3.4, and the definition of $\cos\theta$ it follows that

$$\cos^2 \theta_k \geq (\frac{1+(\sigma/\mu)}{1-(\sigma/2\mu)})^2 \| g_k \|^2 / \| d_k \|^2 . \tag{33}$$

Since $\| g_k \|$ is bounded above in l, we obtain from (33):

$$\cos^2\theta_k \begin{cases} \geq c3/k & \text{if (30) is true} \\ = 1 & \text{if (31) is true} \end{cases}$$

which clearly implies that the series $\sum_k \cos^2\theta_k$ diverges. Now we contradict this result using the line search conditions.

If Ω is an upper bound on $\| \nabla^2 f(x) \|$ where $x \in l$ then we have

$$g_{k+1}^T d_k \leq g_k^T d_k + \Omega \alpha_k \| d_k \|^2 .$$

Thus by using condition (IV) we obtain

$$\alpha_k \geq -\frac{(1-\sigma)}{\Omega \| d_k \|^2} g_k^T d_k$$

which, on substitution into the condition (I) and using the descent property with the definition of $\cos\theta$, leads us to

$$f_{k+1} \leq f_k - K \| g_k \|^2 \cos^2\theta_k,$$

where $K = \rho(1-\sigma)/\Omega > 0$. Since $f(x)$ is bounded on the level set l, $\sum_k \| g_k \|^2 \cos^2\theta_k$ is convergent. Because $\| g_k \|$ is bounded below, this contardicts that $\sum_k \cos^2\theta_k$ is divergent as established above. Since this contradiction arises from our assumption that (13) is not true, the limit (13) must be achieved. □

CHAPTER 4

NUMERICAL RESULTS AND CONCLUSIONS

In this chapter we shall see the effects of inexact line search on the conjugate gradient methods explained in the last chapter. We shall test the algorithms described there on some well known test functions. For the line search we shall use the algorithm described in section 2.4. We used BASIC language to code these algorithms on the computer. It may be noted that the programs we used in our work are not available with K.F.U.P.M computer facilities. A list of these programs may be found in the appendix.

4.1 PARAMETER SETTING

In this section we shall point out how to set the parameters used in the algorithms. Then we shall discuss the effect of different values of these parameters on the performance of the C-G algorithms.

In the line search algorithm the parameter σ plays an important role in the accuracy of the line search. From condition (IV) we have

$$| g_{k+1}^T d_k | \leq -\sigma g_k^T d_k.$$

Thus σ controls the length of the interval of acceptable points. In fact, a small value of σ gives a fairly accurate line search while a value of $\sigma = 0.9$ is used for a weak line search. In the global convergence theorems we have restricted the choice of σ to the interval $(0, 1/2)$ but for the sake of comparison we will choose four different values of σ, i.e., 0.1, 0.4, 0.6, 0.9. In these cases the parameter ρ which appears in conditon (I), has a small effect on the line search since we have $\rho < \sigma$. Therefore, $\rho = 0.01$ will be treated as a typical value in all cases.

Another important factor that affects the performance of the line search algorithm is the initial interval. This interval must be given at hand to start the line search algorithm. Therefore, if this interval is too wide then more iterations will have to be done to locate an acceptable point. In the case of having an approximate value $\overline{\phi}$ of the lower bound of ϕ, we can restrict our search to the interval $(0,\omega)$, where

$$\omega = \frac{\overline{\phi} - \phi(0)}{\rho\phi'(0)}$$

is the point at which the ρ-line intersects the line $\phi = \overline{\phi}$.[1]

Another way to solve this problem is to start with a small interval and then increase it as needed. Since α is positive for the C-G methods, the

initial interval $(0,b)$ must bracket a subinterval of of acceptable points if $\phi(0) < \phi(b)$. Otherwise we increase the value of b. This suggests the following replacement of Step 1 in the Modified Quadratic Algorithm (cf. p. 20):

Step 1. (choice of initial interval (a_1, b_1))

(i) Choose $a_1 = 0$

(ii) Do for $n = 1,2,3,\ldots$

$d_n = a_1 + 0.1n$

$c_n = d_n/2$

If $\phi(0) > \phi(c_n)$ and $\phi(c_n) < \phi(d_n)$ go to (iii).

Else continue.

(iii) Set $b_1 = d_n$ and $c_1 = c_n$.

Go to Step 2 of modified quadratic algorithm.

Shanno restart method uses a positive parameter τ in testing when to restart the C-G method. But in order to prove the convergence of the method we must take $\tau < 1$. Thus in the numerical testing, the value of $\tau = .01$ is adequate in most cases.

Touati Ahmed and Storey method uses two scalars, λ and μ. Although different values of these scalars work better on different types of problems, the algorithm Hybrid 3 achieves its overall best results when $\lambda = 10^{-8}$ and

$\mu = 0.1$ [16]. Since we have to ensure that $1/2 > \mu > \sigma$ in the proof of the convergence theorem of this method, therefore changing the value of σ in the numerical testing forces us to change the value of μ to satisfy the requirement $\mu \in (\sigma, 1/2)$.

4.2 NUMERICAL RESULTS

In this section we shall discuss the effect of inexact line search to different test functions. The various figures appearing in the tables given in section 4.2.2-4.2.3 represent the total number of iterations needed by each algorithm to achieve an 8-digit accuracy as well as the average number of iterations needed by the line search subroutine to locate a point satisfying conditions (I) and (IV) (cf. 2.3).

It may be noted that in performing one iteration of the C-G algorithm for different methods described in the last chapter, the following computations are involved:

- one function evaluation.
- two gradient evaluations each one depending on n equations.

Thus, a total of $2n + 1$ equations were evaluated at each iteration of C-G

algorithm.

On the other hand, computations involved in one step of the line search algorithm are as follows:

- 4 function evaluations,
- one gradient evaluation depending on n equations.

(This step will be carried out only when the point where gradient is evluated satisfies condition (I).)

Thererfore, a total of $n+4$ equations were considered at each iteration of the line search algorithm.

The following abbreviations are used in the tables appearing in 4.2.1-4.2.2.

FR = Fletcher-Reeves method.

PR = Polak-Ribiere method.

SH = Shanno restart method.

TS = Touati Ahmed-Storey method.

NI = Total number of iterations in the main program.

AL = Average number of iterations in the line search subroutine per iteration in the main program.

Blank boxes in the table reflect that no convergence occurs for the corre-

sponding method.

4.2.1 Extended Rosenbrock Test Function

The first nonquadratic function we consider for testing is the Rosenbrock's banana-shaped valley function. The general form of the function in n variables where n is even is given as follows:

$$f = \sum_{i=1}^{n/2} [100(x_{2i} - x_{2i-1}^2)^2 + (1 - x_{2i-1})^2].$$

We shall test this function for $n = 2, 4, 10, 20$.

For $n = 2$, we choose the starting point to be $(-1.2, 1)$, whereas our choice for the higher dimensions will be as follows: $x = (\underbrace{-1.2, 1}, \underbrace{-1.2, 1}, \ldots, \underbrace{-1.2, 1})$.

Rosenbrock for n=2

Method	$\sigma = 0.1$		$\sigma = 0.4$		$\sigma = 0.6$		$\sigma = 0.9$	
	NI	AL	NI	AL	NI	AL	NI	AL
FR	70	2.3	90	2.4	87	1	57	3.3
FR (reset)	37	10	36	6.2	38	2	37	1.9
PR	9	60						
PR (reset)	14	30						
SH (FR)	35	12.6	58	6	58	5	41	6
SH (PR)	9	60						
TS	20	20	23	13	26	9		

It can be seen from the above table that Shanno restart for the Polak-Ribiere method shows a high improvement over the usual restart. Also, while the Fletcher-Reeves method when restarted after n iterations is better to the same method without restart, Polak-Ribiere method is vice-versa.

Rosenbrock n=4

Method	$\sigma = 0.1$		$\sigma = 0.4$		$\sigma = 0.6$		$\sigma = 0.9$	
	NI	AL	NI	AL	NI	AL	NI	AL
FR	15	24	51	6	23	10	24	9
FR (reset)	17	22	23	13	34	7	32	7
PR	11	7						
PR (reset)	13	6						
SH (FR)	34	11	34	9	21	11	65	5.3
SH (PR)								
TS	18	24	33	10		21	16	

It is clear from the above two tables that the Polak-Ribiere method is not globally convergent when inexact line search is used.

Rosenbrock n=10

Method	$\sigma = 0.1$		$\sigma = 0.4$		$\sigma = 0.6$		$\sigma = 0.9$	
	NI	AL	NI	AL	NI	AL	NI	AL
FR	18	4.4	19	4.4				
FR (reset)	18	4.2	28	3.5	28	5.6		
PR	12	6.5						
PR (reset)	12	6.5						
SH (FR)	18	4.4	19	4.4				
SH (PR)	14	5.6						
TS	19	5.4	20	3	24	3.5	33	2.8

Touati Ahmed and Storey method involves less number of line searches when we reduce the accuracy as noted in the above table.

Rosenbrock n=20

Method	$\sigma = 0.1$		$\sigma = 0.4$		$\sigma = 0.6$		$\sigma = 0.9$	
	NI	AL	NI	AL	NI	AL	NI	AL
FR	21	5.7	48	3.5				
FR (reset)	21	5.7	43	4	28	3.7		
PR	11	4.8						
PR (reset)	11	4.8						
SH (FR)	21	4.5	62	3.5				
SH (PR)	12	7.2						
TS	16	4	55	1.6	30	5		

It may be noted in the above table that the convergence is achieved within a number of steps less than the dimension of the search space.

4.2.2 Powell Test Function

The function defined by

$$f = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4$$

was devised by Powell as a test function for minimization algorithms . Its Hessian is a singular matrix of rank 2 at its minimum point $x^* = 0$ and

is positive definite elsewhere. The starting point is $(3,-1,0,1)$ for all the methods reflected in the followimg table:

Method	$\sigma = 0.1$		$\sigma = 0.4$		$\sigma = 0.6$		$\sigma = 0.9$	
	NI	AL	NI	AL	NI	AL	NI	AL
FR	159	1	356	1	44	2	111	2
FR (reset)	63	2	68	2	48	2	48	2
PR	57	2	57	2		-		
PR (reset)	68	2	37	2				
SH (FR)	274	1						
SH (PR)	360	1						
TS	38	1	186	1				

In this table we can see a significant improvement due to the Touati Ahmed-Storey method over all other methods.

4.2.3 Last Example

To end this subsection we test the algorithms on the following function

$$f = \frac{1}{4}x_1^4 - 2x_1x_2 + x_1 + x_2^2$$

where the minimum point is $x^* = (-1.618, -1.618)$. The starting point is $(0,0)$ for all the methods shown in the following table:

Method	$\sigma = 0.1$		$\sigma = 0.4$		$\sigma = 0.6$		$\sigma = 0.9$	
	NI	AL	NI	AL	NI	AL	NI	AL
FR	59	4.1	59	2.4	89	3	92	3.1
FR (reset)	8	2.4	9	1.6	10	1.3	10	1.3
PR	7	2.1	6	1.3	6	1.2	6	1.2
PR (reset)	7	2.1	6	1.3	6	1.2	6	1.2
SH (FR)	59	4.1	59	2.5	89	3	92	3.1
SH (PR)	7	2.1	6	1.3	6	1.2	6	1.2
TS	7	2.1	5	1.4	5	1.2	5	1.2

It may be noted from the above table that the Shanno restart method for the Fletcher-Reeves algorithm does not show any improvement over the usual restart. We can also see that all the Polak-Ribiere algorithms perform the same for this problem.

4.3 CONCLUDING REMARK

It is clear from the above tables that reducing the accuracy in the line search leads to a smaller number of iterations in the line search subroutine but increases the number of iterations in the main program with certain exceptions. There are some cases where reducing the accuracy in line search gives a better result.

In our observation, the more exact line search we perform, the more effort we will have to put to reach the optimum of the objective function. More precisely, we refer the reader to the table "Rosenbrock for n=2" for FR(reset)-method. One can note that for $\sigma = 0.1$(the case when inexact line search is close enough to the exact one), we need a total of $37 \times 10 = 370$ number of line searches as compared to the case of $\sigma = 0.6$ where we need only $38 \times 2 = 76$ number of line searches. In return, we pay the price of only one extra iteration (i.e. instead of performing 37 iterations for $\sigma = 0.1$, we perform 38 iterations for $\sigma = 0.6$).

As pointed out earlier, $2n + 1$ equations per iteration are involved in the main program of each method, while we need $n+4$ equations to be considered in the line search. Thus, for large n, we encounter large number of equations in the main program as compared to the number of equations involved in the

line search subroutine. This amounts to more effort in the main program for large n.

As expected, Polak-Ribiere method is the most affected method by inexact line search. That is due to the fact that the descent property is not satisfied in all iterations if inexact line search is used. Also, we note that restarting the Polak-Ribiere method after every n iterations does not provide any improvement over the method without restart if inexact line search is being used.

Since the increase in the value of σ implies the increase in μ which affects the convergence in theorem 3.5, Touati Ahmed and Storey method gives no convergence when the value of σ is more than 0.6. On the other hand, Touati Ahmed-Storey method is believed to be the best method over all other if the objective function becomes more nonlinear as can be seen in the Powell test function.

REFERENCES

1. Al-Baali , M. and Fletcher ,R. , "An efficient line search for nonlinear least squares " , Journal of Optimzation theory and Appl. Vol. 48, 359-377, 1986 .

2. Al-Baali , M. "Descent property and global convergence of the Fletcher-Reeves method with inexact line search" , IMA Journal of Numer. Analy. vol. 5, 121-124, 1985.

3. Dixon, L.C.W.,Spedicato,E.,and Szego,G.P.,Nonlinear Optimization Theory and Algorithms,Birkhauser,Boston 1980.

4. Fletcher,R.,Practical Method of Optimization ,2nd edition,Wiley, London, 1987 .

5. Fletcher,R.,and Reeves,C.M.,"Function minimization by conjugate gradient", Computer Journal,Vol. 7 ,149-154 , 1964.

6. Goldstein,A.A.,"On steepest descent", SIAM Journal on Control,Vol. 3(1), 147-151, 1965.

7. Hestenes,M.R.,Conjugate Direction Methods in Optimization, Springer-verlag,Berlin 1985.

8. Horst,R.,and Tuy,H.,Global Optimization ,Springer-Verlag,Berlin, (1990).

9. Luenberger,D.G.,Linear and Nonlinear Programming,2nd edition, Addison Wesley, 1984.

10. McKeown,J.J.,Meegan,D.,and Sprevak,D.,An Introduction to Unconstrained Optimization , Adam Hilger , London, 1990.

11. Minoux,M.,Mathematical Programming , John Wiley and Sons, London, 1986.

12. Powell,M.J.D.,Convergence properties of algorithms for nonlinear optimization,Report no. DAMTP 1985/NA1,Department of Applied Math. and Theor. Physics, University of Cambridge,England 1985.

13. Powell,M.J.D.,Nonconvex minimization calculation and the conjugate Ggadient method, Report no. DAMTP 1983/NA14 ,Department of Applied Math. and Theor. Physics,University of Cambridge,England 1983.

14. Scales,L.E.,Introduction to Nonlinear Optimization,Macmillan , London 1985 .

15. Shanno,D.F.,"Globally convergent conjugate gradient algorithm", Mathematical Programming,vol. 33, 61-67, 1985 .

16. Touati-Ahmed, and Storey,C.,"Efficient hybrid conjuget gradient techniques" , Journal of Optimization Theory and Appl. ,Vol. 64(2), , 379-397, 1990.

17. Wolfe, P.,"Convergence conditions for ascent method ",SIAM rev., vol. 11, 226-235, 1968.

APPENDIX

```
10 REM                 **********************************
30 REM                 * QUADRATIC FIT                   *
50 REM                 **********************************
60 REM
70   A(1)=0   :B(1)=1   :   C(1)=.3   :   I=0
80   FA=   EXP(-A(1))+A(1)^2
90   FB=   EXP(-B(1))+B(1)^2
100   FC=   EXP(-C(1))+C(1)^2
101   LPRINT   "X=";C(1),"F=";FC
120 I=I+1
121  IF  I=6  THEN  END
130    ALPHA=(1/2)*(((B(I)^2-C(I)^2)*FA+(C(I)^2-A(I)^2)*FB+
(A(1)^2-B(1)^2)*FC)/((B(1)-C(I))*FA+(C(I)-A(I))*FB+(A(I)-B(I))*FC     ))
140  FX=EXP(-ALPHA)+ALPHA^2
150   LPRINT   "X=";ALPHA,"F=";FX
240  IF  (ALPHA  <  C(I))AND(FX  <  FC)  THEN  250  ELSE  260
250    A(I+1)=A(I):B(I+1)=C(I):C(I+1)=ALPHA:FB=FC:FC=FX:GOTO    120
260  IF  (ALPHA  >  C(I))AND(FX  >  FC)  THEN  270  ELSE  280
270    A(I+1)=A(I):B(I+1)=ALPHA:C(I+1)=C(I):FB=FX:GOTO    120
280  IF  (ALPHA  <  C(I))AND(FX  >  FC)  THEN  290  ELSE  300
290    A(I+1)=    ALPHA:B(I+1)=B(I):C(I+1)=C(I):FA=FX:GOTO    120
300    A(I+1)=C(I):B(I+1)=B(I):C(I+1)=ALPHA:FA=FC:FC=FX:GOTO    120
```

```
X=  .3              F=   .8308182
X=    .3618209      F=   .8273215
X=    .3528194      F=   .8271856
X=    .3517377      F=   .827184
X=    .3519035      F=   .8271841
X=    .351227       F=   .8271845
```

```
1 REM                  ********************************
2 REM                  *  NEWTON  '  S  METHOD  *
3 REM                  ********************************
4 REM
5 REM
10  INPUT " enter initial point " ,X
11   F=EXP(-X)+X^2
15   PRINT  "x=";X,"F=";F
16  FOR I=1 TO 3
20  DF=  -  EXP(-X) +2 * X
30   DF2=  EXP(-X)+2
40   X=X-(DF/DF2)
45   F=EXP(-X)+X^2
50   PRINT  "x=";X,"F=";F
60 NEXT I
```

```
x= .3            F=   .8308182
x=   .3513782    F=   .8271842
x=   .3517337    F=   .8271841
x=   .3517337    F=   .827184
```

```
10 REM              ********************************
20 REM              * CONJUGATE GRADIENT METHOD *
30 REM              * (Fletcher-Revees)          *
40 REM              *            And             *
50 REM              * (Polak-Ribiere )           *
60 REM              ********************************
70   DIM   A(10)   ,B(10),C(10),X(10),G(10),GG(10)
80  COUNT=0
90  PRINT" Enter the diminsion of the problem"
100 INPUT N
110 PRINT" Enter your starting point"
120 FOR I=1 TO N
130  INPUT X(I) , "Reading the initial point "
140 NEXT I
150 F = (THE DEFINITION OF THE FUNCTION)
160 IF F <= LF THEN END ( LF IS THE ACCURACY WANTED )
170 FOR I=1 TO N
180 G(I) = DEFINITION OF THE GRADIENT VECTOR
190 NEXT I
200 FOR I= 1 TO N
210  D(I)=  -  G(I)
220 NEXT I
230  REM **** starting the cycle **** -
240 FOR J= 1 TO N
250  REM *** calling the line search subroutine ***
260 GOSUB (LINE SEARCH )
270 FOR I=1 TO N
280 X(I) = X(I) + ALPHA * D(I)
290 NEXT I
300 F= ( DEFINITION OF THE FUNCTION )
310 FOR I=1 TO N : PRINT X(I) , F : NEXT I
320 IF (F < LF ) THEN END        " termination "
330 FOR I =1 TO N
340  GG(I)=(DEFINTION OF THE GRADIENT)" next gradient vector "
350 NEXT I
360 NORM(G) = 0 : NORM(GG) =0
370 FOR I=1 TO N
380   NORM(G)=NORM(G)+G(I)^2
390   NORM(GG)=NORM(GG)+GG(I)^2
400 NEXT I
410 REM For the Fletcher-Revees method we will use step 470
420 REM and for the Polak-Ribiere method we use step 480
430 REM
440 BETA = NORM(GG)/NORM(G)
450 BETA = ((GG(I) -G(I) ) * GG(I) )/( NORM(G) )
460 REM
470 FOR I = 1 TO N
480 D(I)= -GG(I) + BETA * D(I)
490   G(I)=GG(I)
500 NEXT I
510 NEXT J
520 GOTO 200          " Reset the method "
```

```
10 REM              ********************************
20 REM              *       SHANNO RESTART         *
30 REM              *         METHOD               *
40 REM              ********************************
50   DIM   X(10)   ,G(1),GG(I)
60  J=0  :  T=.01
70 PRINT " ENTER THE DIMINSION OF THE PROBLEM "
80 INPUT N
90 PRINT " ENTER THE INITIAL POINT "
100 FOR I = 1 TO N
110 INPUT X(I) ,          " READING THE INITIAL POINT "
120  NEXT I
130  INPUT  X(I)
140 F= ( THE DEFINITION OF THE FUNCTION )
150 IF F <= LF THEN END
160 G(1)= ( THE DEFINITION OF THE GRADIENT VECTOR )
170  SUM  =  1/(G(1)^2)
180 FOR I = 1 TO N
190   D(1)=-G(1)
200 NEXT I
210 J=J+1
220  REM  ****** calling the subroutine ******
230  GOSUB ( LINE SEARCH )
240 FOR I = 1 TO N
250  X(I)=X(I)+  ALPHA  *  D(I)
260 NEXT I
270 F= ( THE DEFINITION OF THE FUNCTION )
280 FOR I = TO N
290 PRINT  X(I) , F
300 IF (F < LF) THEN END
310 FOR I= 1 TO N
320 GG(I) = ( THE DEFINITION OF THE GRADIENT VECTOR )
330 NEXT I
340  SUM  =SUM  +  (1/(GG(1)^2))
350  GAMA  =  T/((GG(I)^2)*  SUM)
360 BETA= ( THE FORMULA OF BETA )
370 FOR I = 1 TO N
380  DD(1)=-GG(1)  +BETA  *  D(1)
390 NEXT I
400   THETA   =   (GG(I)*DD(I))^2/((GG(I)^2)*(DD(I)^2))
410 IF (THETA >= GAMA ) THEN 420 ELSE 440
420 D(1)=DD(1)          : GOTO 450
430 FOR I = 1 TO N
440   D(1)=-GG(I)
450   G(1)=GG(I)
460 NEXT I
470 GOTO 210
```

```
10 REM              ***************************************
20 REM              * TOUATI-AHMED AND STOREY              *
30 REM              *            METHOD                    *
40 REM              ***************************************
50   DIM   X(I),G(I),GG(I)
60  MU=.3 : LM = 1E-08 : J=0
70  PRINT " enter the diminsion of the problem "
80  INPUT N
90  PRINT " enter the initial point "
100  FOR I= 1 TO N
110 INPUT X(I) ,                " Reading the initial point"
120  NEXT I
130  INPUT  X(I)
140 F=( THE DEFINITION OF THE FUNCTION )
150 IF F <= LF THEN END
160  FOR I= 1 TO N
170  G(I) = ( THE GRADIENT VECTOR )
180  D(I)  =  -  G(I)
190  NEXT I
200 J=J+1
210  REM  ***** calling the Line search subroutine ******
220  GOSUB ( LINE SEARCH )
230  FOR I=1 TO N
240   X(I)=X(I)+  ALPHA  *D(I)
250 F= (THE DEFINITION OF THE FUNCTION )
260  FOR I= 1 TO N
270  PRINT  X(I) , F
280  NEXT I
290  IF (F < LF) THEN END
300  FOR I = 1 TO N
310  GG(I)= ( THE DEFINITION OF THE GRADIENT VECTOR )
320  NEXT I
330  NORM(GG) = 0 : NORM(G)= 0
340 BFR=( FLETCHER-REVEES FORMULA FOR BETA )
350  BPR=( POLAK-RIBIERE FORMULA FOR BETA )
360   REM  ****  Choosing  Beta  *******
370  FOR I= 1 TO N
380   NORM(GG)=GG(I)^2+NORM  (GG)
390  NORM(G)  =  G(I)^2  +  NORM(G)
400  NEXT I
410  IF  (LM  *  NORM(GG)  <=  (2*MU(J+1))  THEN  430  ELSE  420
420  BETA  =  0 : GOTO 490
430  IF  (  BPR  <  0  )  THEN  440  ELSE  450
440  BETA  =  BFR : GOTO 490
450  IF  (BPR  <=  ((1/(2*MU))*(NORM(GG))/(NORM(G)))  THEN  460  ELSE  470
460  BETA  =  BPR : GOTO 490
470  BETA  =  BFR
480  FOR  I  =1  TO  N
490   D(I)=-GG(I)  +  BETA  *  D(I)
500   G(I)   =GG(I)
510  NEXT  I
520  GOTO  200
```

```
10 REM              ********************************
20 REM              *  LINE  SEARCH  SUBROUTINE  *
30 REM              ********************************
40 REM
50  A(1)=0  :B(1)=1  :  C(1)=.3  :  I=0  :  RO=.05  :  SE=.2
60 FA= EVALUATE THE FUNCTION AT A(1)
70 FB= EVALUATE THE FUNCTION AT B(1)
80 FC= EVALUATE THE FUNCTION AT C(1)
90 DF= DERIVATIVE OF THE FUNCTION AT ALPHA = 0
100 I=I+1
110    ALPHA=(1/2)*(((B(I)^2-C(I)^2)*FA+(C(I)^2-A(I)^2)*FB+
(A(I)^2-B(I)^2)*FC)/((B(I)-C(I))*FA+(C(I)-A(I))*FB+(A(I)-B(I))*FC))
120 FOR I = 1 TO N
130  XX(I)=X(I)  +  ALPHA  *  D(I)
140 NEXT I
150 FX= EVALUATE THE FUNCTION AT ALPHA
160 IF (FX <= (F+ALPHA*RO*DF )) THEN 180 ELSE 220
170 FOR I = 1 TO N
180 GA(I) = GRADIENT VECTOR
190 NEXT I
200 DFA= DERIVATIVE OF THE FUNCTION AT THE NEW POINT
210 IF (ABS(DFA) <= (-1* SE*DF)) THEN RETURN
220 IF (ALPHA < C(I)) AND (FX < FC) THEN 230 ELSE 240
230  A(I+1)=A(I)  :  B(I+1)=C(I)  :  C(I+1)=ALPHA
    : FB=FC : FC=FX : GOTO 100
240 IF (ALPHA > C(I)) AND (FX > FC) THEN 250 ELSE 260
250   A(I+1)=A(1):B(1+1)=ALPHA:C(I+1)=C(1):FB=FX:GOTO   100
260 IF (ALPHA < C(I)) AND (FX > FC) THEN 270 ELSE 280
270  A(1+1)=  ALPHA:B(I+1)=B(I):C(I+1)=C(I):FA=FX:GOTO   100
280   A(I+1)=C(I):B(I+1)=B(I):C(I+1)=ALPHA:FA=FC:FC=FX:GOTO   100
```